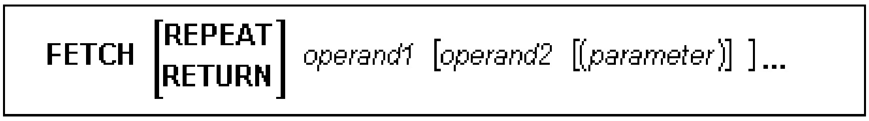
FETCH



```
FETCH [REPEAT] operand1 [operand2 [(parameter)] ]...
      [RETURN]
```

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1	C	S				A												yes	no
Operand2	C	S	A	G		A	N	P	I	F	B	D	T	L		G		yes	yes

Related Statements: CALLNAT | PERFORM

Function

The FETCH statement is used to execute a Natural object program written as a main program. The program to be loaded must have been previously stored in the Natural system file with a CATALOG or STOW command. Execution of the FETCH statement does not overwrite any source program in the Natural source work area.

REPEAT

REPEAT causes Natural to suppress the prompt for user input for each INPUT statement issued during the execution of the FETCHed program. It may be used to send information about the execution of the program to the terminal without the user having to reply with ENTER.

RETURN

Without the specification of RETURN, the execution of the program issuing the FETCH statement will be terminated immediately and the FETCHed program will be activated as a *main program* (level 1).

If a program is invoked with FETCH RETURN, the execution of the invoking program will be suspended - not terminated - and the FETCHed program will be activated as a *subordinate program* on a higher level. Control is returned to the invoking program when an END or ESCAPE ROUTINE statement is encountered in the FETCHed program. Processing is continued with the statement following the FETCH RETURN statement.

With FETCH RETURN, you invoke and execute an object of type program as a routine.

Program Name - operand1

The name of the program module (maximum 8 characters) can be specified as an alphanumeric constant or the content of an alphanumeric variable of length 1 to 8.

Natural will attempt to locate the program in the library currently active at the time the FETCH is issued. If the program is not found, Natural will attempt to locate the program in the steplibs. If the program is still not found, an error message will be issued.

The program name may contain an ampersand (&); at execution time, this character will be replaced by the current value of the system variable *LANGUAGE. This makes it possible, for example, to invoke different programs for the processing of input, depending on the language in which input is provided.

Parameters - operand2

The FETCH statement may also be used to pass parameter fields to the invoked program. A parameter field may be defined with any format. The parameters are converted to a format suitable for a corresponding INPUT field. All parameters are placed on the top of the Natural stack.

The parameter fields can be read by the FETCHed program using an INPUT statement. The first INPUT statement will result in the insertion of all parameter field values into the fields specified in the INPUT statement. The INPUT statement must have the sign specification (SG=ON) for parameter fields defined with numeric format, because each parameter field defined with numeric format in the FETCH statement will receive a sign position if its value is negative.

If more parameters are passed than are read by the next INPUT statement, the extra parameters are ignored. The number of parameters may be obtained with the Natural system variable *DATA.

Note:
If operand2 is a time variable (format T), only the time component of the variable content is passed, but not the date component.

parameter

If *operand2* is a date variable, you can specify the session parameter DF as *parameter* for this variable. The session parameter DF is described in the Natural Reference documentation.

Additional Considerations

In addition to the parameters passed explicitly with FETCH, the FETCHed program also has access to the established global data area.

The FETCH statement may cause the internal execution of an END TRANSACTION statement based on the setting of the Natural profile parameter OPRB as set by the Natural administrator. If a logical transaction is to span multiple Natural programs, the Natural administrator should be consulted to ensure that the OPRB parameter is set correctly.

Example

Invoking Program:

```
/* EXAMPLE 'FETEX1': FETCH
/**********************************************************
DEFINE DATA LOCAL
1 #PNUM (A8)
1 #FNC  (A1)
END-DEFINE
/**********************************************************
INPUT 10X 'SELECTION MENU FOR EMPLOYEES SYSTEM' /
      10X '-' (54) //
      10X 'ADD  '     '(A)' /
      10X 'PURGE'     '(P)' /
      10X 'UPDATE'    '(U)' /
      10X 'TERMINATE' '(.)' //
      10X 'PERSONNEL NUMBER:' #PNUM ///
      10X 'PLEASE ENTER FUNCTION: ' #FNC
/**********************************************************
DECIDE ON EVERY VALUE OF #FNC
  VALUE 'A'
    FETCH 'ADD-RT' #PNUM
  VALUE 'P'
    FETCH 'PUR-RT' #PNUM
  VALUE 'U'
    FETCH 'UPD-RT' #PNUM
  VALUE '.'
    STOP
  NONE
    REINPUT 'PLEASE ENTER A VALID FUNCTION' MARK *#FNC
END-DECIDE
/**********************************************************
END
```

Invoked Program:

```
/* EXAMPLE 'PUR-RT' (PROGRAM FETCHED IN EXAMPLE 'FETEX1')
/*******************************************************
DEFINE DATA LOCAL
1 #PERS-NR (A8)
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
END-DEFINE
/*******************************************************
INPUT #PERS-NR
/*******************************************************
FIND NUMBER EMPLOY-VIEW WITH PERSONNEL-ID = #PERS-NR
 IF *NUMBER = 0
  WRITE NOTITLE 'NO RECORD FOUND'
  STOP
 END-IF
/*******************************************************
 FIND EMPLOY-VIEW WITH PERSONNEL-ID = #PERS-NR
  DELETE
  END TRANSACTION
  WRITE NOTITLE 'RECORD DELETED'
 END-FIND
/*******************************************************
END
```

```
          SELECTION MENU FOR EMPLOYEES SYSTEM
          -----------------------------------------------------

          ADD    (A)
          PURGE (P)
          UPDATE (U)
          TERMINATE (.)

          PERSONNEL NUMBER: 1150304


          PLEASE ENTER FUNCTION:  P
```

```
RECORD DELETED
```